VIEWS Technical User Information

A Getting Started Guide to Using the Validations and Edits Web Service

Table of Contents

Change Date	Changed By	Reason
2/11/2011	John Brown	Updated. Added additional info relevant to the pilot extension.
4/11/2011	John Brown	Added some Surveillance component and Medical edit specific information.
9/8/2011	Doug Ringley	Added references to Rare Cause, Ill Defined / Trivial Cause and Site Check validations. Added Testing Guidelines section.
9/15/2011	Doug Ringley	Added additional test scenarios.

Introduction

The Validations and Edits Web Service that is currently in development is being provided in order to move more of the edits and corrections to mortality data to the front of the process at the point of data entry. This should provide more complete and accurate data and improve throughput, as well as reduce the workload caused by the need to revisit records late in the process.

A prototype web service was shown at NAPSIS in June 2010 to demonstrate how the system might work once complete. Currently the project is entering the pilot phase, where vendors and states will be given access to the current web service and information regarding how to communicate with the service and use the data provided. The current version of the service is now running and publicly accessible. It will provide:

- a means for vendors and states to set up and test communication with the web service
- a way to validate the XML messages to be sent to the service
- a way to receive a sample return XML message so that they can also begin the process of determining the effort required to use the information provided by the service in the respective systems

The VIEWS web service is a Microsoft WCF web service, and can be found at https://www.vitalsviewsvalidation.com/validationservice.svc. This pilot phase web service is being provided so that vendors and states can have a means to begin the analysis of how they might use the service within their own products as we work towards a production system. Since this is not a production system, performance and service level guarantees cannot be made at this time, however every effort will be made to provide advance notice if it should be necessary to take the service down temporarily. It should also be noted that since this is not a production system, security measures are not fully in place to meet production standards and this pilot service is to be used only for testing and verification purposes. It should not be used to send actual mortality data across the internet.

Accessing the Web Service

Methods of accessing the web service will vary depending on what technology your current application is written in. If your current application is already accessing a web service you should have no problem connecting to VIEWS. If you should encounter any difficulties accessing the service, be sure to include pertinent information in your request for help such as programming language, hosting environment, etc.

Once you have created a connection to the web service, your application can send a message string to the service and receive a return message string. The web service will accept the text you send and attempt to validate the XML it receives. Whenever possible, if the XML is invalid the service will return an informational message regarding the validation error encountered. If the text received cannot be validated to the extent required to send a useful message, a more generic informational message will be sent to let you know that your message failed validation. If the message you send to the service can be validated, the service will send you a return message regarding data errors. The current service will simply send a static message back that should allow you to begin the analysis of how you will use the

XML the service returns. When development moves forward the service will begin to do live validation of the data you send.

The pilot phase of the project will perform the following types of validations:

- Mortality focused spellchecking
- Rare word identification
- Abbreviation validation
- Data Validations, including field length, data type, boundary validation, dependency rules and ICD code determination.
- Medical Edits
- Surveillance
- Rare Cause
- Ill Defined / Trivial Cause
- Site Check

XML Schema

The XML schema is provided for you in a separate document available from the VIEWS project manager. This document also includes sample XML text that might be sent to and received from the VIEWS service. The XML schema will provide you with enough information to create your own XML and validate the messages before you send them.

The XML schema should provide you with enough information to understand what parts of the mortality record the service needs to perform validations. In addition you can also gather information such as data type, data length, and boundaries for particular fields.

FAQ

What is the ID attribute in the Certificate XML? Do I need it?

The ID attribute is a way for the calling system to identify the requests for validation and match them to the responses from the web system. If you choose to make multiple asynchronous requests to VIEWS you may send ID=1, ID=2 and ID=3 simultaneously. When you handle the response event, the XML returned will identify the responses with the IDs you sent in the request. This enables you to match the correct response to the correct data record. Note that VIEWS does not use this information for anything, so if you want to you can set the ID to 42 or 999999 or whatever you want to use as a default. VIEWS will still accept the record. Note that under no circumstances should you ever use the certificate number or any other identifying information as the ID.

What fields are validated?

	Line 1a	Line 1b	Line 1c	Line 1d	Duration 1a	Duration 1b	Duration 1c	Duration 1d	Line 2	Place Of Injury	Injury Description	Date Fields	Transport
Spelling	X	X	X	X					X	X	X		
Abbreviations	X	X	X	X					X		X		
Data Validations – Boundary Check												X	
Data Validations – ICD check	X	X	X	X	X	X	X	X	X	X	X		X
Data Validations – non keyboard characters	X	X	X	X	X	X	X	X	X	X	X		X
Ill Defined / Trivial Cause	X	X	X	X					X		X		
Medical Edits	X	X	X	X					X		X		
Rare Words	X	X	X	X					X		X		
Surveillance	X	X	X	X					X		X		
Rare Cause	X	X	X	X					X		X		
Rare Words	X	X	X	X					X		X		
Site Check	X	X	X	X					X		X		

Table 1: Validations By Field

How is Hour of Injury data validated?

The hour of injury field must be 0000-2359, 9999, or blank. This is enforced by the xml schema. Further validations are done on the server side to check that the hour of injury and injury time code "fit" together for lack of a better phrasing. Time code must be A, P, M or blank (again, enforced by the xml schema), but when added to the hour of injury field it must make sense. As an example, you can send 0300 and P as 3:00pm, or 1500 and M as 3:00pm. If you tried to send 1500 and P (or A for that matter) you would get an error message. To summarize, hour of injury and injury time code must create a valid time when combined, unless one of them is blank.

How does abbreviation validation work?

Abbreviations covers two different types of abbreviations, known acceptable abbreviations and known ambiguous abbreviations. Known acceptable abbreviations are those that are accepted by the CDC in that a particular abbreviation is commonly used and accepted as corresponding to a single term or phrase, such as AAA - Abdominal Aortic Aneurysm. A known ambiguous abbreviation is defined as a known abbreviation that is commonly used in more than one way, such as RA - Renal Artery, Right Arm, Rheumatoid Arthritis. Known acceptable abbreviations should pass through the system without any action. An ambiguous abbreviation should return a message along with a suggested "did you mean this" prompt list. NOTE: Future system will include a way for you to force abbreviations to warn on known acceptable abbreviations as well as ambiguous abbreviations.

What data validations are done?

In addition to the ICD, boundary and non-keyboard character validations, many other validations are done on many of the fields passed to the service. Much of this work is done simply by enforcing the XML schema. For example, many boundary and field length checks are enforced by the XML schema. Certain date validations must be performed on the service side to check for date dependency and formatting violations. Date of death is checked to determine if any fields are blank or undefined. If the date appears to be a complete date then the date is tested for validity (no February 30th, etc.). Date of injury goes through the same checks as date of death, and then if both dates are valid dates a check is made to verify that the injury occurred on or before the date of death. Date of surgery goes through the same checks as date of injury. Additionally, surgery date is also checked to see if all date components are listed as undefined.

What is a non-keyboard character?

Ç è ¥ ▓ Φ

Why are there so many different messages returned for each validation? Don't I only need one? Do I need to display all 3?

The message levels are basically intended to provide different levels of information based on user roles. In the end it is really up to vendors and states how they choose to define roles, which messages to use, whether or not to implement our messages or create your own. Many times the messages will be the same for all levels. To summarize it though, level 1 messages are basic messages for data entry type staff who would not have the training or the authority to make some decisions, but who would be capable of spelling corrections and verifying the accuracy of the information they entered compared to the information they have been given. Level 3 messages are intended for trained staff that could make decisions regarding the correctness of the information. As an example, a data entry person might not have the information they need to determine the actual condition behind an ambiguous abbreviation, they would only be able to verify that the abbreviation they entered was the one on the form they are entering. A doctor on the other hand would know if RA was rheumatoid arthritis or renal artery. The messages should always be in order by level, but generally you will just be pulling a single message out based on user role. For something like a spell checker it wouldn't really matter, but for a rare cause issue you would have different prompts based on whether the user was a doctor who could make decisions or a data entry staff member who could only verify the information they were given.

Why do I receive an invalid XML message instead of validation information?

The information message is the generic message that will be returned if the XML cannot be validated. If the service receives invalid XML it will attempt to return information about why the XML is invalid, but processing won't continue.

How much information do I have to send to the service to get validation feedback?

You can make as many calls as you want with whatever information you want to send. The service will validate whatever information you send and ignore anything you don't include. If you were to call the service 3 times with text in Line A you would just get Line A validations back each time.

Testing Guidelines

In order to perform basic testing, you may want to create test XML packets like the following example. This will allow the developer to test submitting a well-formed XML stream to the VIEWS service to ensure that the interaction with the service is working properly. Refer to the XML Schema document for specific information regarding XML messages to and from the VIEWS service.

Example of XML to send to the VIEWS service:

```
<?xml version='1.0' encoding='utf-8' ?>
<Certificate Year='2000' State='NC' ID='012345' xmlns='WebMMDS'>
        <YearOfDeath>2000</YearOfDeath>
        <MonthOfDeath>02</MonthOfDeath>
        <DayOfDeath>20</DayOfDeath>
        <Sex>F</Sex>
        <AgeUnits>1</AgeUnits>
        <Age>033</Age>
        <Line1a>ABANDONNED</Line1a>
        <Line1b>ABANDONNED</Line1b>
        <Duration1b>ABANDONNED</Duration1b>
        <Line1c>ABANDONNED</Line1c>
        <Duration1c>ABANDONNED</Duration1c>
        <Line1d>ABANDONNED</Line1d>
        <Duration1d>ABANDONNED</Duration1d>
        <Line2>ABANDONNED</Line2>
        <TobaccoUse>U</TobaccoUse>
        <Pregnancy>1</Pregnancy>
        <MannerOfDeath>N</MannerOfDeath>
        <MonthOfInjury>01</MonthOfInjury>
        <DayOfInjury>02</DayOfInjury>
        <YearOfInjury>2000</YearOfInjury>
        <HourOfInjury>2300</HourOfInjury>
        <InjuryTimeCode>M</InjuryTimeCode>
        <PlaceOfInjury>ABANDONNED</PlaceOfInjury>
        <WorkInjury>N</WorkInjury>
        <InjuryDescription>ABANDONNED</InjuryDescription>
        <Transport>ABANDONNED</Transport>
        <Autopsy>N</Autopsy>
        <AutopsyFindings>N</AutopsyFindings>
        <MonthOfSurgery>02</MonthOfSurgery>
        <DayOfSurgery>02</DayOfSurgery>
        <YearOfSurgery>2000</YearOfSurgery>
        <Activity>1</Activity>
</Certificate>
```

Example of VIEWS output XML:

Based upon the previous input example, upon successful submission of the input packet, the following output packet would be returned:

```
<?xml version="1.0" encoding="utf-8"?>

<ReturnMessage Year="2000" State="NC" ID="12345" xmlns="WebMMDS">
        <ValidationData type="IllDefined" messageid="IllDefined" field="Line1a">
                <Term>Certificate contains ill-defined and/or trivial terms.</Term>
        </ValidationData>
        <ValidationData type="Spelling" field="Line1a">
                <Term>ABANDONNED</Term>
                <Suggestion rank="1">ABANDONED</Suggestion>
        </ValidationData>
        <ValidationData type="Spelling" field="Line1b">
                <Term>ABANDONNED</Term>
                <Suggestion rank="1">ABANDONED</Suggestion>
        </ValidationData>
        <ValidationData type="Spelling" field="Line1c">
                <Term>ABANDONNED</Term>
                <Suggestion rank="1">ABANDONED</Suggestion>
        </ValidationData>
        <ValidationData type="Spelling" field="Line1d">
                <Term>ABANDONNED</Term>
                <Suggestion rank="1">ABANDONED</Suggestion>
        </ValidationData>
        <ValidationData type="Spelling" field="Line2">
                <Term>ABANDONNED</Term>
                <Suggestion rank="1">ABANDONED</Suggestion>
        </ValidationData>
        <ValidationData type="Spelling" field="InjuryDescription">
                <Term>ABANDONNED</Term>
                <Suggestion rank="1">ABANDONED</Suggestion>
        </ValidationData>
      <ValidationData type="Spelling" field="PlaceOfInjury">
                <Term>ABANDONNED</Term>
                <Suggestion rank="1">ABANDONED</Suggestion>
        </ValidationData>
</ReturnMessage>
```

Sample Verification Examples

If you want to verify that your application is correctly connected to VIEWS and that your interface is displaying messages for each validation type, you can use the following information to submit to VIEWS. Each of the items below should cause a response from VIEWS.

Validation Test	*Line 1a*	*Age (in years)*
Spelling	ABANDONNED	N/A
Data Validations	X90	N/A
Abbreviations	RA	N/A
Rare Words	FRIENDLY	N/A
Rare Causes	BOTULISM	N/A
Ill-Defined/Trivial Causes	CARDIAC ARREST	N/A
Surveillance	H1N1	N/A
Medical Edits	Tetanus neonatorum	2